BigARTM: Open Source Library for Regularized Multimodal Topic Modeling of Large Collections

Konstantin Vorontsov[1,3], Oleksandr Frei[2], Murat Apishev[3], and Peter Romov[4]

[1] Department of Intelligent Systems at Dorodnicyn Computing Centre of RAS,
Moscow Institute of Physics and Technology, `voron@forecsys.ru`
[2] Schlumberger ...
[3] Moscow State University, ...
[4] Yandex ...

Abstract. BigARTM

Keywords: probabilistic topic model, Probabilistic Latent Sematic Analysis, Latent Dirichlet Allocation, Additive Regularization of Topic Models, stochastic matrix factorization, EM-algorithm, BigARTM.

1 Introduction

Topic modeling is a rapidly developing branch of statistical text analysis [?]. Topic model reveals a hidden thematic structure of the text collection and finds a compressed representation of each document by a set of its topics. Such models appear to be highly useful for many applications including information retrieval for long-text queries, revealing research trends and research fronts, classification, categorization, summarization and segmentation of texts. More ideas, models and applications are outlined in the survey [?].

From the statistical point of view, a probabilistic topic model defines each topic by a multinomial distribution over words, and then describes each document with a multinomial distribution over topics. From the optimizational point of view, topic modeling can be considered as a special case of constrained approximate stochastic matrix factorization. Learning a factorized representation of a text collection is an ill-posed problem, which has an infinite set of solutions. Then a regularization must be used to impose problem-oriented restrictions and thus to choose a more reasonable solution.

Hundreds of models adapted to different situations are described in the literature. For practitioners, most of them are too difficult to quickly understand, compare, combine and embed into applications. Then there was a common practice to use only the simplest models such as *Probabilistic Latent Semantic Analysis*, PLSA [?] and *Latent Dirichlet Allocation*, LDA [?]. Some of the difficulties are rooted in Bayesian learning framework, which is dominating approach in topic modeling. Bayesian inference of each kind of model is a unique and laborious mathematical work, which prevents the unification and the flexible manipulation of various topic models. Until now, there was no freely available development tools to easily design, modify, select, and combine topic models.

In this paper we announce **the BigARTM open source project** for parallel distributed online topic modeling of large collections, `http://bigartm.org`. The theory of BigARTM is based on non-Bayesian multicriteria approach — *Additive Regularization of Topic Models*, ARTM [**?**]. In ARTM a topic model is learned by maximizing the weighted sum of the log-likelihood and additional regularization criteria. The optimization problem is solved by a general regularized expectation-maximization (EM) algorithm, in which any regularization criteria or their combination can be substituted. Many known Bayesian topic models were revisited in terms of ARTM in [**?**,**?**]. Compared to the Bayesian approach, ARTM makes topic models easier to design, infer, combine and explain, thus reducing barriers to entry into topic modeling research field.

The rest of the paper is organized as follows. In section 2 we introduce a multimodal topic model generalized for documents with multiple accompanying discrete metadata. In section 3 we describe a fast online algorithm [**?**] generalized for any additively regularized topic models. In section 4 we consider the parallel architecture and implementation details of BigARTM library. In section 5 we report some experiments on large datasets. In section 6 we discuss advantages, limitations and open problems of BigARTM.

2 Multimodal regularized topic model

Let D denote a finite set (collection) of texts and W^1 denote a finite set (vocabulary) of all terms from these texts. Each term can be a single word or a key phrase. Denote n_{dw} the number of times the term w appears in the document d. A document can contain not only words, but also elements of other modalities, which we also call terms. Each j-th modality is defined by the finite set (vocabulary) of its terms W^j, $j = 1, \ldots, m$. Examples of not-word modalities are: authors, class or category labels, date-time stamps, references to or from other documents, entities mentioned in texts, objects found in the images illustrating texts, users that read or downloaded documents, advertising banners, etc.

Following the ideas of Correspondence LDA [**?**] and Dependency LDA [**?**] we introduce the topic model of each j-th modality:

$$p(w \,|\, d) = \sum_{t \in T} p(w \,|\, t)\, p(t \,|\, d) = \sum_{t \in T} \phi_{wt}\theta_{td}, \quad d \in D,\ w \in W^j,\ j = 1, \ldots, m.$$

The model parameters $\phi_{wt} = p(w \,|\, t)$ and $\theta_{td} = p(t \,|\, d)$ form the matrixes $\Phi^j = \left(\phi_{wt}\right)_{W^j \times T}$ of *term probabilities for the topics*, and $\Theta = \left(\theta_{td}\right)_{T \times D}$ of *topic probabilities for the documents*. Matrixes Φ^j, if stacked vertically, form $W \times T$-matrix Φ, where $W = W^1 \sqcup \cdots \sqcup W^m$ is disjoint union of all modalities. Matrices Φ^j and Θ are *stochastic*, that is, their columns ϕ_t^j, θ_d are non-negative and normalized representing discrete distributions. The number of topics $|T|$ is usually assumed to be much smaller than $|D|$ and some of modalities $|W^j|$.

To learn parameters Φ^j, Θ from the multimodal text collection we maximize the log-likelihood for j-th modality:

$$\mathcal{L}_j(\Phi^j, \Theta) = \sum_{d \in D} \sum_{w \in W^j} n_{dw} \ln p(w \mid d) \to \max_{\Phi^j, \Theta},$$

where n_{dw} is the number of occurrences of the term $w \in W^j$ in the document d. Note that topic distributions of documents Θ are common for all modalities. Following the ARTM approach, we add a regularization penalty term $R(\Phi, \Theta)$ and solve the constrained multicriteria optimization problem via scalarization:

$$\sum_{j=1}^{m} \tau_j \mathcal{L}_j(\Phi^j, \Theta) + R(\Phi, \Theta) \to \max_{\Phi, \Theta}; \tag{1}$$

$$\sum_{w \in W^j} \phi_{wt} = 1, \ \phi_{wt} \geq 0; \qquad \sum_{t \in T} \theta_{td} = 1, \ \theta_{td} \geq 0. \tag{2}$$

Theorem 1. *If the function $R(\Phi, \Theta)$ is continuously differentiable then the local maximum (Φ, Θ) of the problem (1), (2) satisfies the system of equations with auxiliary variables $p_{tdw} = p(t \mid d, w)$:*

$$p_{tdw} \propto \phi_{wt} \theta_{td};$$

$$\phi_{wt} \propto \left(n_{wt} + \phi_{wt} \frac{\partial R}{\partial \phi_{wt}} \right)_+; \qquad n_{wt} = \sum_{d \in D} n_{dw} p_{tdw};$$

$$\theta_{td} \propto \left(\sum_{j=1}^{m} \tau_j n_{td}^j + \theta_{td} \frac{\partial R}{\partial \theta_{td}} \right)_+; \qquad n_{td}^j = \sum_{w \in W^j} n_{dw} p_{tdw}.$$

This statement follows from Karush–Kuhn–Tucker conditions. The system of equations can be solved by various numerical methods. Particularly, the simple-iteration method is equivalent to the EM algorithm, which is typically used in practice. Note that with one modality ($m = 1$) this theorem gives E-step and M-step equations of the regularized EM algorithm in ARTM [?,?]. With no regularization ($R = 0$) it corresponds to probabilistic latent semantic analysis [?].

3   Online topic modeling

Following the ideas of Online LDA [?]

4   BigARTM architecture

4.1   Algorithm

Algorithm 1 Online EM-algorithm

1: Initialize ϕ_{wt} for all $w \in W$ and $t \in T$;
2: $n_{wt} := 0$, $\tilde{n}_{wt} := 0$
3: **for all** batches D_j, j = 1,...,J **do**
4:     $\tilde{n}_{wt} := \tilde{n}_{wt} + ProcessBatch(D_j, \phi_{wt})$
5:     **if** (synchronize) **then**
6:         $n_{wt} := Merge(n_{wt}, \tilde{n}_{dw})$;
7:         $r_{wt} := Regularize(n_{wt})$;
8:         $\phi_{wt} := Normalize(n_{wt} + r_{wt})$;
9:         $\tilde{n}_{dw} := 0$;

Algorithm 2 $ProcessBatch(D_j, \phi_{wt})$

Require: Batch D_j, Matrix ϕ_{wt};
Ensure: n_{wt};
1: $n_{wt} := 0$ for all $w \in W$ and $t \in T$;
2: **for all** $d \in D_j$ **do**
3:     initialize θ_{td} for all $t \in T$;
4:     **repeat**
5:         $Z_w := \sum_{t \in T} \phi_{wt}\theta_{td}$ for all $w \in d$;
6:         $\theta_{td} := \frac{1}{n_d}\sum_{w \in d} n_{dw}\phi_{wt}\theta_{td}/Z_w$ for all $t \in T$;
7:     **until** θ_d converges;
8:     increment $\tilde{n}_{wt}$ by $n_{dw}\phi_{wt}\theta_{td}/Z_w$ for all $w \in d$ and $t \in T$;

4.2 Concurrency

BigARTM can utilize multiple CPU cores by processing several batches in parallel, which follows the rule of expressing parallelism at the highest possible level. A alternative would be to parallelize the $ProcessBatch(D_j, \phi_{wt})$ routine, but keeping it simple has many obvious benefits — single-threaded code is much easier to modify and optimize.
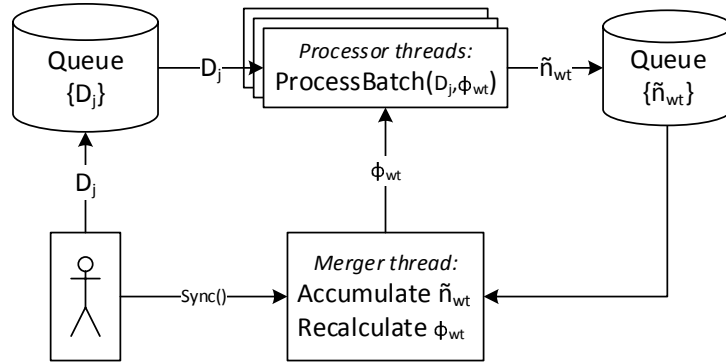

Fig. 1. Diagram of key BigARTM components

Inputs and the outputs of $ProcessBatch$ routine are stored in two in-memory queues, locked for push and pop operations with spin locks. This approach does not add any noticeable synchronization overhead because both queues only store smart pointers to the actual data objects, so push and pop operations does not involve copying of real memory.

Smart pointers are also essential for handling of the ϕ_{wt} matrix. This matrix is *read* by all processors threads, and can be *written* at any time by the merger thread. To resolve this conflict we keep two copies of the ϕ_{wt} — an *active Φ* and a *background Φ* matrices. The active matrix is read-only, and is used by the processor threads. The background matrix is being built in a background by the merger thread from n_{wt} and r_{wt} counters, and once it is ready merger thread marks it as active. Before processing a new batch the processor threads gets the current active matrix from the merger thread. This object is passed via shared smart pointer to ensure that processor thread can keep ownership of the ϕ_{wt} matrix until the batch is fully processed.

Having two copies of the Φ matrix is an obvious drawback of this solution. An alternative would be to use atomic CPU instructions to read and write values of the Φ matrix. Such operations are very efficient, but still come at a considerable synchronization cost [5]. Using them for all reads and writes of the Φ matrix would cause significant performance degradation for merger and processor threads. Besides, an arbitrary overlap between reads and writes of the Φ matrix eliminates any possibility of producing a deterministic result. The design with two copies of the Φ matrix give much more control over this and in certain cases allows BigARTM to behave in a fully deterministic way.

The design with two Φ matrices only support a single merger thread, and we believe it should cape with all $\hat{n}_{wt}$ updates coming from many threads. This is a reasonable assumption because $Merge$ routine takes only about $O(W * T)$ operations to execute, while $ProcessBatch()$ takes $O(N_{nz} * T * I)$ operations, where W is the size of the dictionary, N_{nz} is the number of non-zero entries in the batch, T is the number of topics, and I is the averate number of inner iterations in $ProcessBatch()$ routine. N_{nz}/W is typically 100 to 1000 (based on datasets in UCI Bag-Of-Words repository), and I is $10\ldots20$, so the ratio safely exceeds the expected number of cores (up to 32 physical CPU cores in modern workstations, and even 60 cores if we are talking about Intel Xeon Phi co-processors).

4.3 Memory layout

4.4 Programming interface

The core of BigARTM is written in C++, but we decided to wrap it into an "extern C" API. This makes it very easy to consume BigARTM from almost any programming environment, because there is always a way of calling "extern C" code (ctypes module for Python, loadlibrary for Matlab, PInvoke for C#, etc).

---

[5] http://stackoverflow.com/questions/2538070/atomic-operation-cost

5 Experiments

6 Conclusions